3D Environmental Audio With OpenAL

OpenAL Overview

Session 205

These are confidential sessions—please refrain from filming or taking pictures.

Graphics and Media

3D Environmental Audio With OpenAL

OpenAL Overview

Elliot Sedegah
DTS Engineer

https://developer.apple.com/wwdc2005

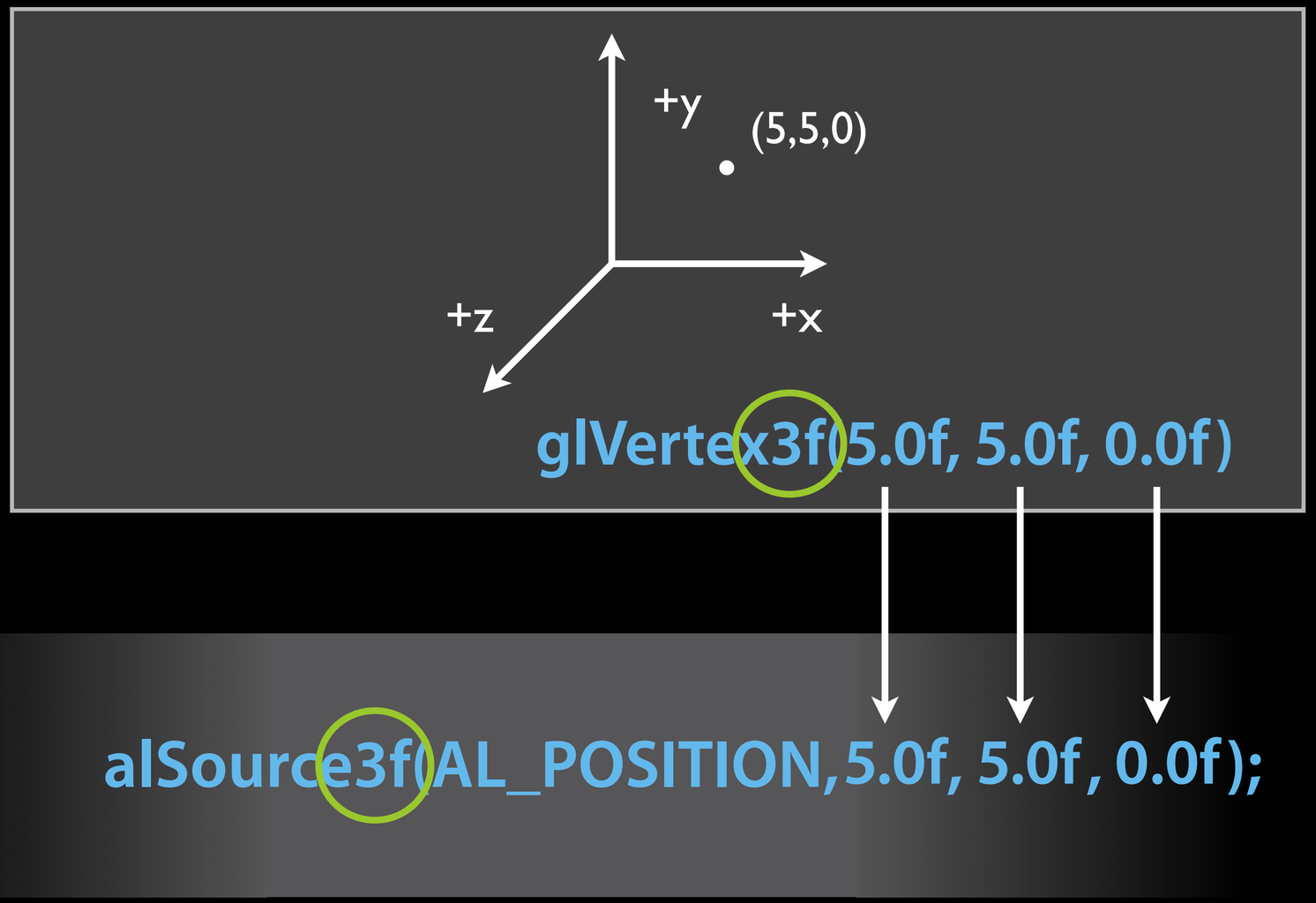
What Is OpenAL?

- Cross Platform 2D and 3D Audio API
- Compliment to OpenGL
- Open Source
  - http://www.openal.org
- Shipping with Tiger as OpenAL.framework
  - Compatible with Mac OS X 10.2.8 or greater

Main Features

- Stereo and Surround Sound
- Positional Audio
- Distance-Based Attenuation
- Pitch Changing

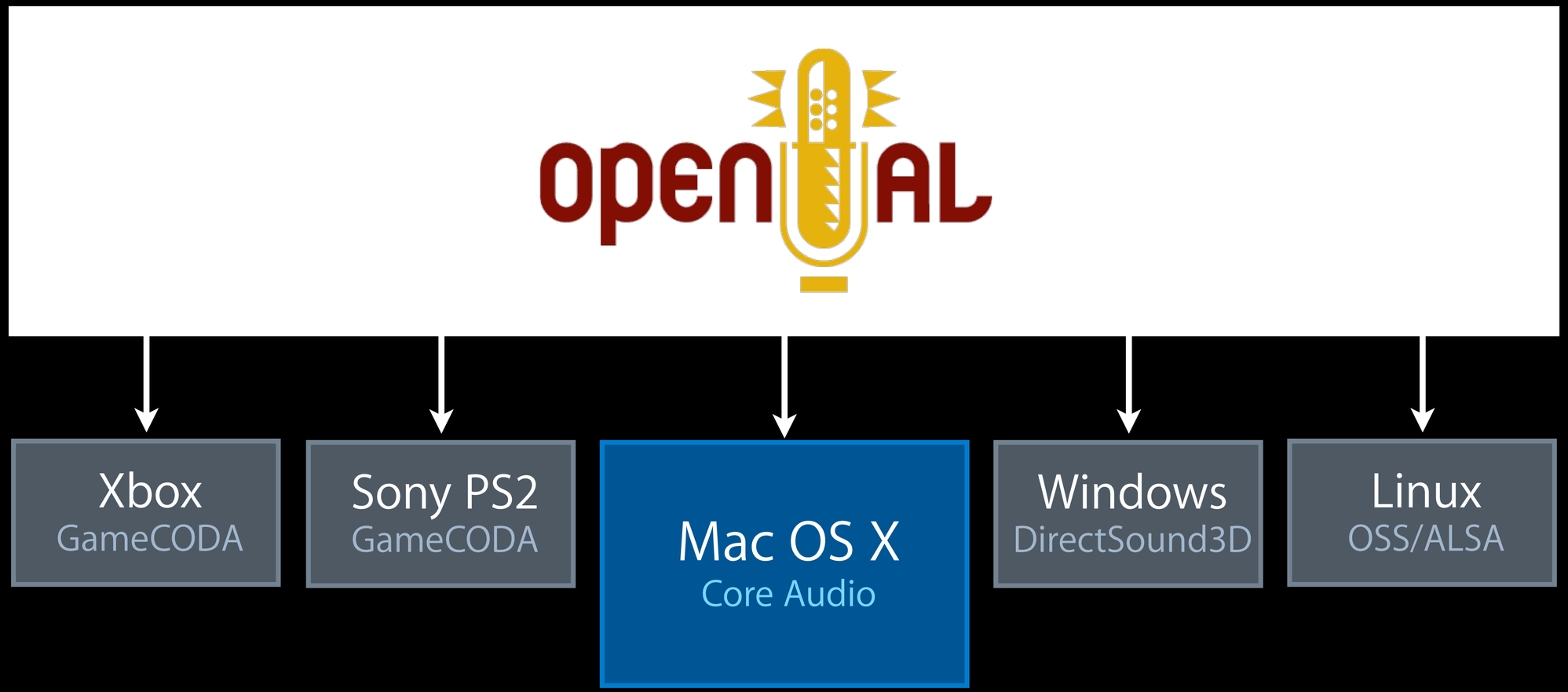
Same Coordinate System as OpenGL



Similar API Conventions

Built With the Power of Core Audio

Cross Platform



For full listing of platforms: http://www.openal.com/platforms.html

Shipped Using OpenAL

	Macintosh	Windows	Linux	FreeBSD	Irx
AlienFlux	●	●	●		
America's Army Operations	●	●	●		
A Tale in the Desert II	●	●	●		●
Blender 3D	●	●	●	●	
Dark Horizons: Lore	●	●	●		
Escape From Monkey Island	●				
Feeding Frenzy	●				
Harry Potter and the Chamber of Secrets	●	●			
Jedi Knight: Jedi Academy	●	●			
Jedi Knight 2	●	●			
Marble Blast	●	●	●		
MegaCorps Online	●	●	●		
Orbz	●	●	●		
Unreal Tournament 2003	●	●	●		
Unreal Tournament 2004	●	●	●		
XPlane	●	●	●		

Game Engines and Libraries

- AgentAE
- C4 Engine
- Lightweight Java Game Library
- Torque Engine
- Unreal Engine

Developer Story

Ryan Gordon
Feeding Frenzy

https://developer.apple.com/wwdc2005

OpenAL Architecture and Mac OS X Implementation

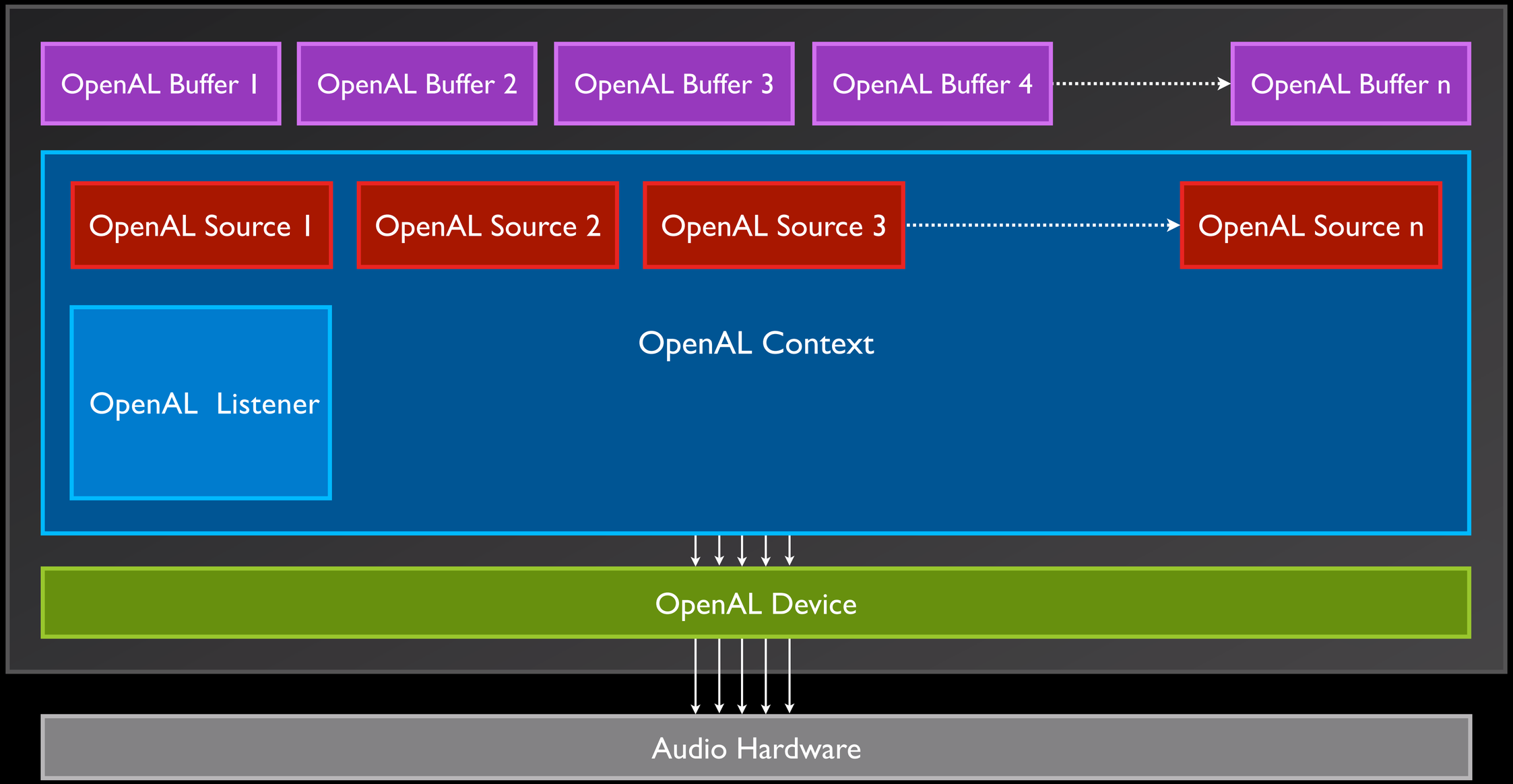
Bob Aron
Core Audio Engineer

https://developer.apple.com/wwdc2005

What You'll Learn

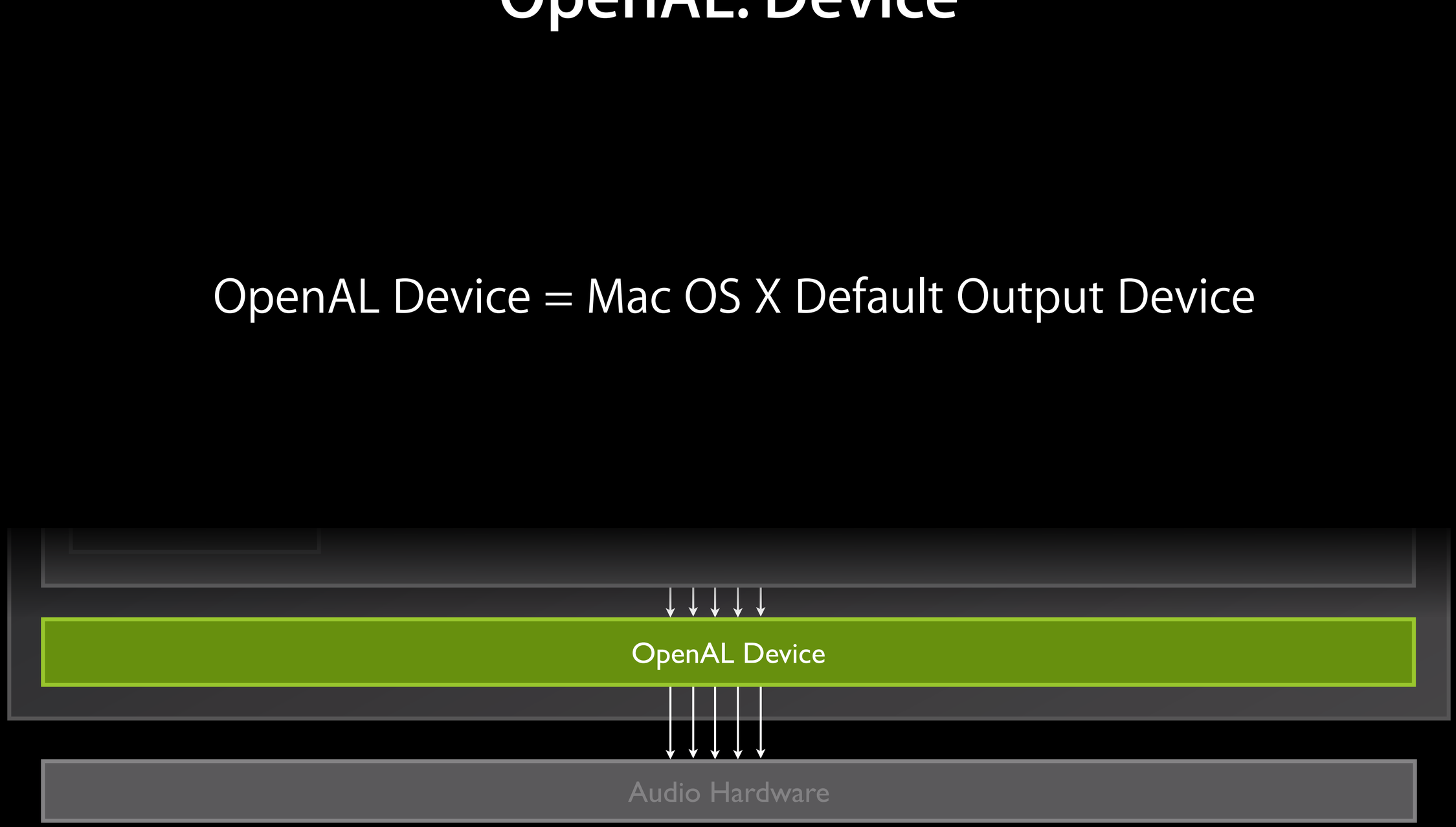
- OpenAL architecture
  - The virtual sound environment
  - Essential APIs
- OpenAL is a powerful audio engine everywhere
  - Difficult spatial rendering work done in the implementations
  - Applications require NO audio hardware knowledge
  - How Core Audio makes OpenAL great on Mac OS X

OpenAL Environment



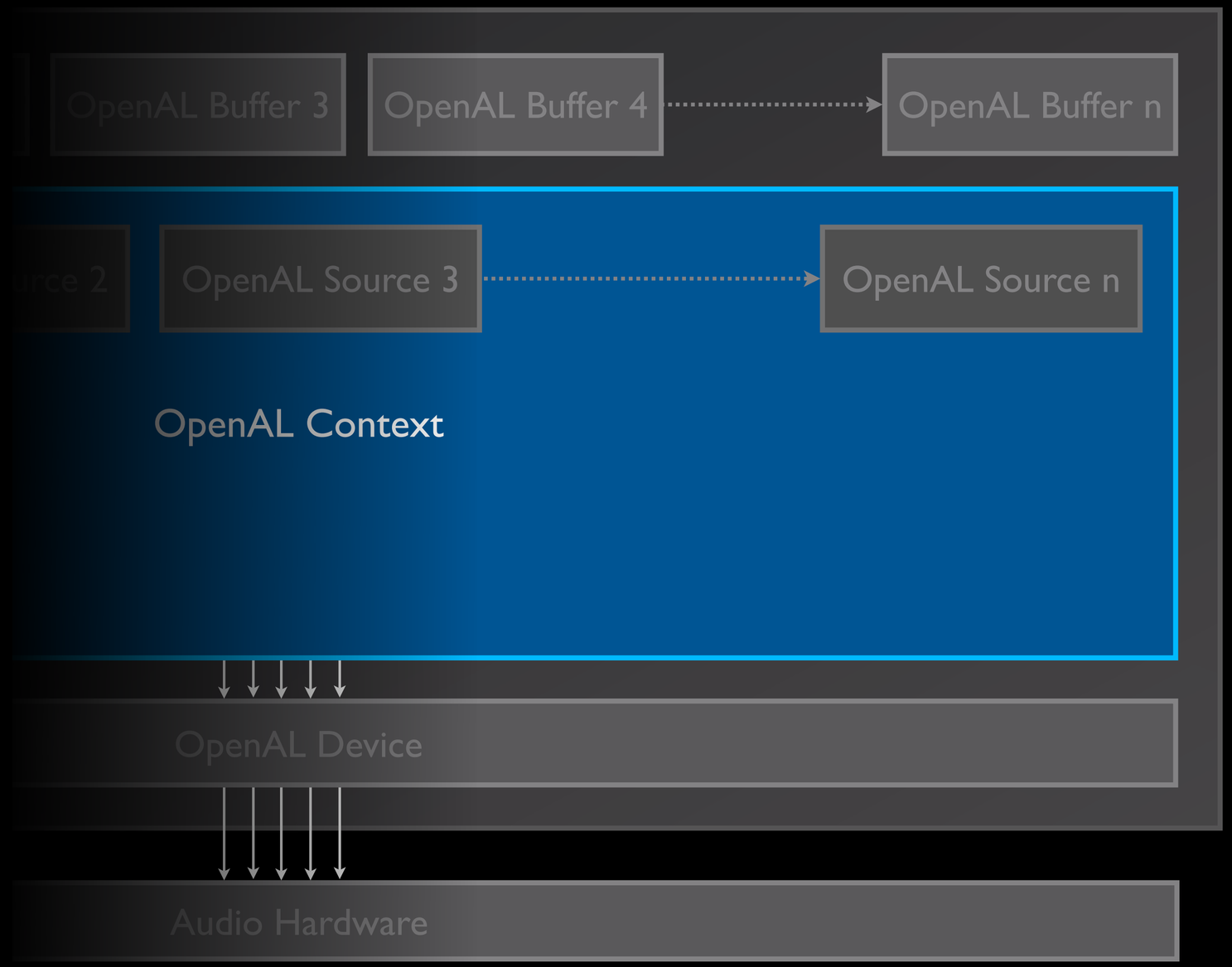
OpenAL: Device

OpenAL Device = Mac OS X Default Output Device



OpenAL: Context

- Objects Making Sound
- Environmental Settings
- Talks to OpenAL Device

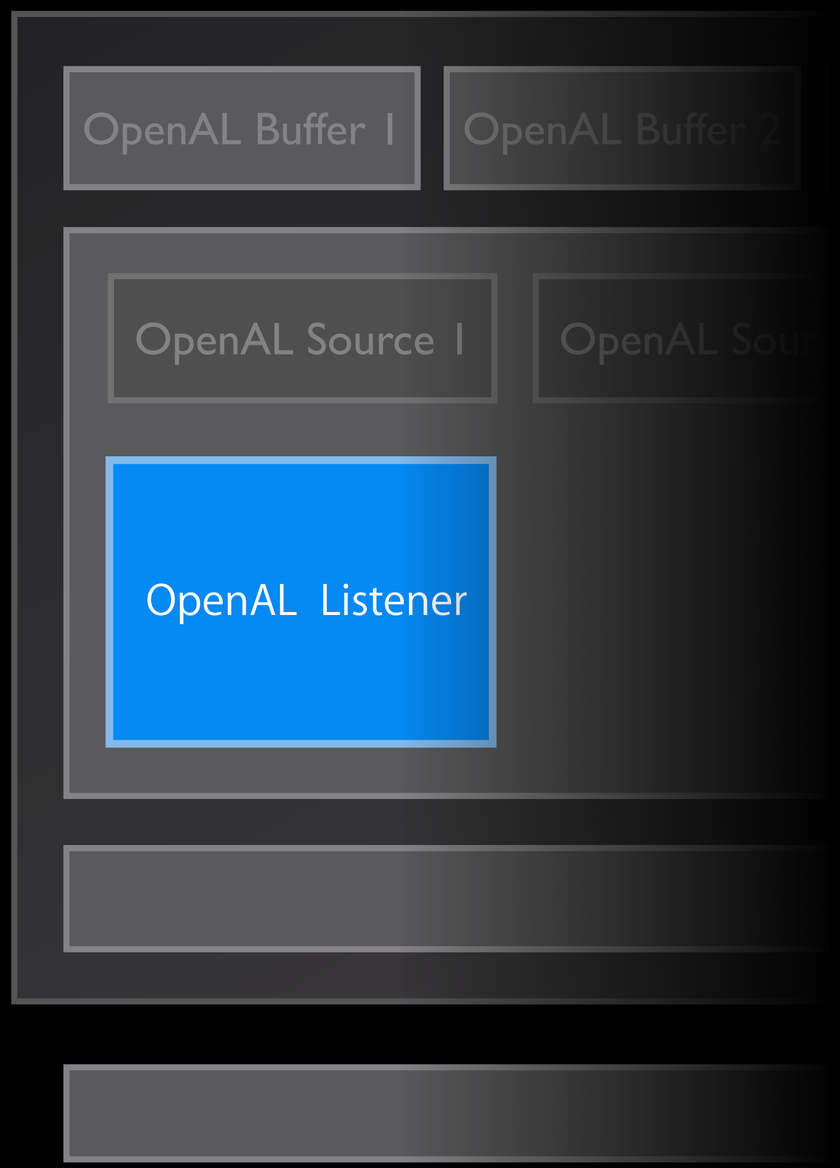


OpenAL: Devices and Contexts

- Opening/Closing an OpenAL Device
  - `ALCdevice* alcOpenDevice(ALCubyte *deviceName)`
  - `ALCvoid   alcCloseDevice(ALCdevice *device)`
- Creating/Destroying an OpenAL Context
  - `ALCcontext*  alcCreateContext( ALCdevice *device, ALCint   *attrList)`
  - `ALCvoid      alcDestroyContext (ALCcontext *context)`
- Making an OpenAL Context ‘Current’
  - `ALCboolean   alcMakeContextCurrent(ALCcontext *context)`

Demo Code: Initializing OpenAL, Part 1

```
void InitializeOpenAL()
{
  // Create a new OpenAL Device using the Default Output Device
  ALCdevice  *newDevice = alcOpenDevice(NULL);
  if (newDevice != NULL)
  {
    // Create a new OpenAL Context
    ALCcontext  *newContext = alcCreateContext(newDevice, 0);
    if (newContext != NULL)
    {
      // Make the new context the Current OpenAL Context
      alcMakeContextCurrent(newContext);
      // Create some OpenAL Source objects
      alGenBuffers(NUM_BUFFERS_SOURCES, gBuffers);
      // Create some OpenAL Buffer objects
      alGenSources(NUM_BUFFERS_SOURCES, gSources);
    }
  }
}
```



OpenAL Listener

- Only ONE Listener per Context
- Positionable in Virtual Space
- The User's Experience of the Virtual Space

OpenAL Listener: Orientation

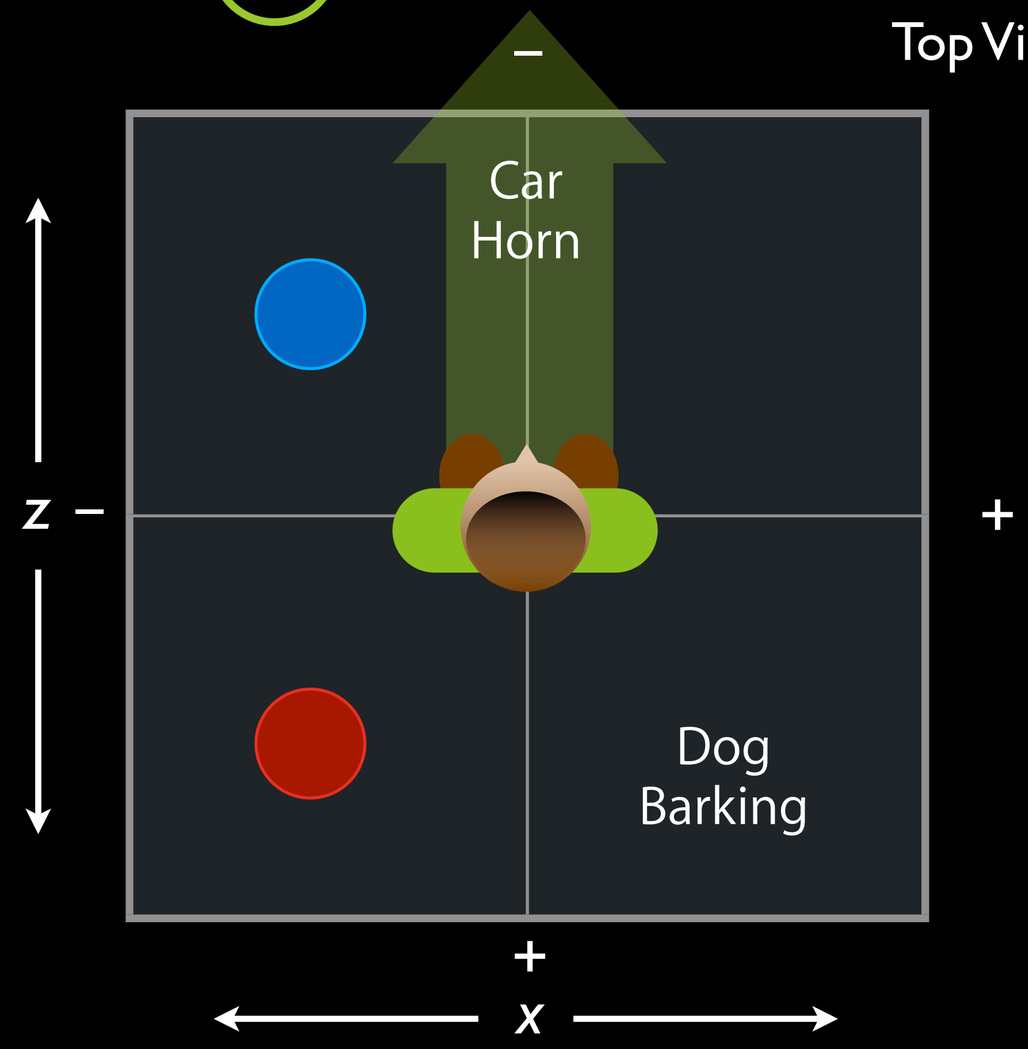
AL_ORIENTATION

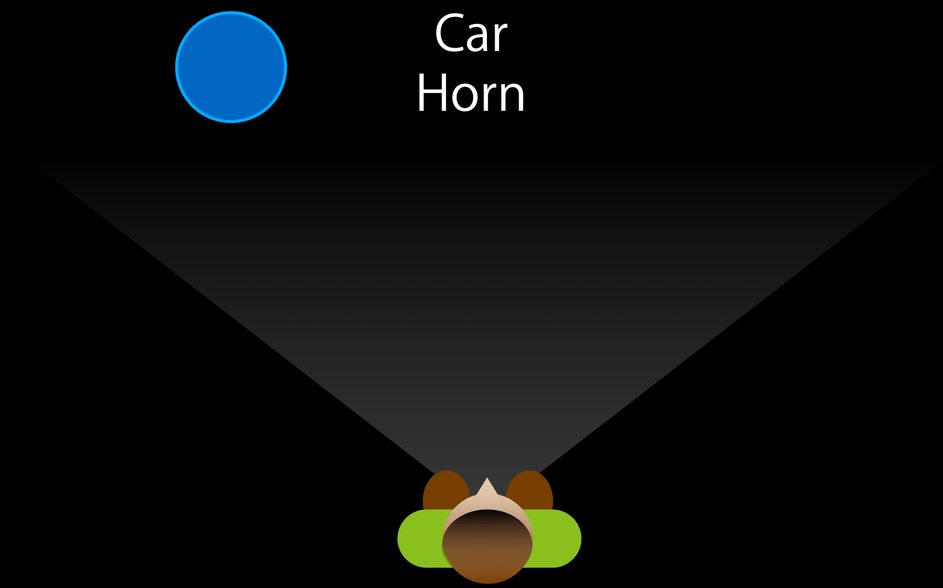
- OpenAL AL_ORIENTATION = OpenGL : gluLookAt()
- Render Audio Appropriately to the User's Speakers
- Described by two Vectors:
  - AT = Direction the Listener is facing
  - UP = Direction pointing up from the top of the Listener's head
  - Vectors are Orthogonal

```
// Orient the Listener Upside Down (-Y) and facing +Z
ALfloat listenerOrientation[6] = { 0.0, 0.0, 1.0,    // AT vector +Z
                                   0.0, -1.0, 0.0}    // UP vector -Y
alListenerfv(AL_ORIENTATION, listenerOrientation);
```

OpenAL Listener: Orientation

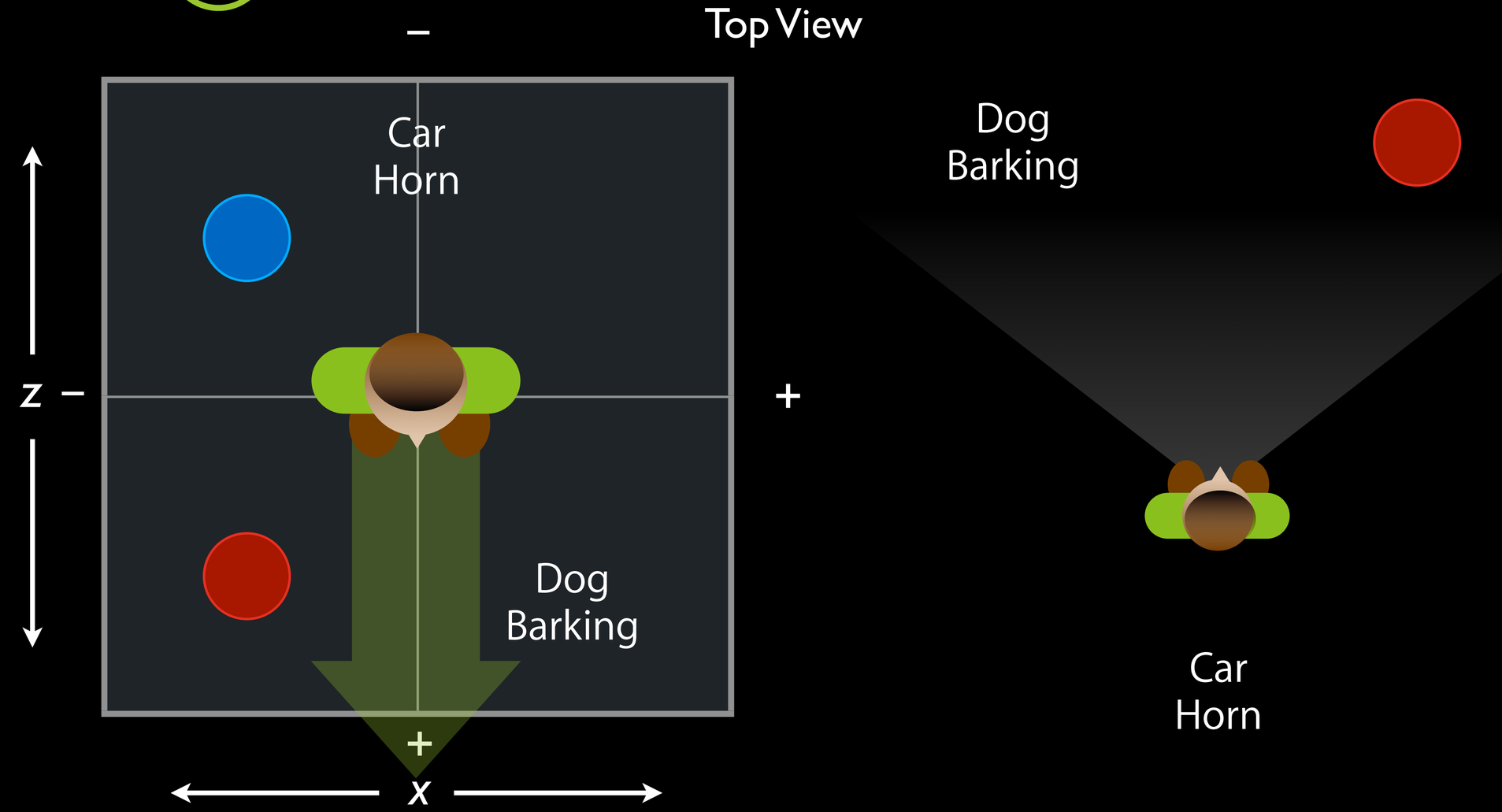
x:0, y:0, z: -1 = Listener facing Negative Z

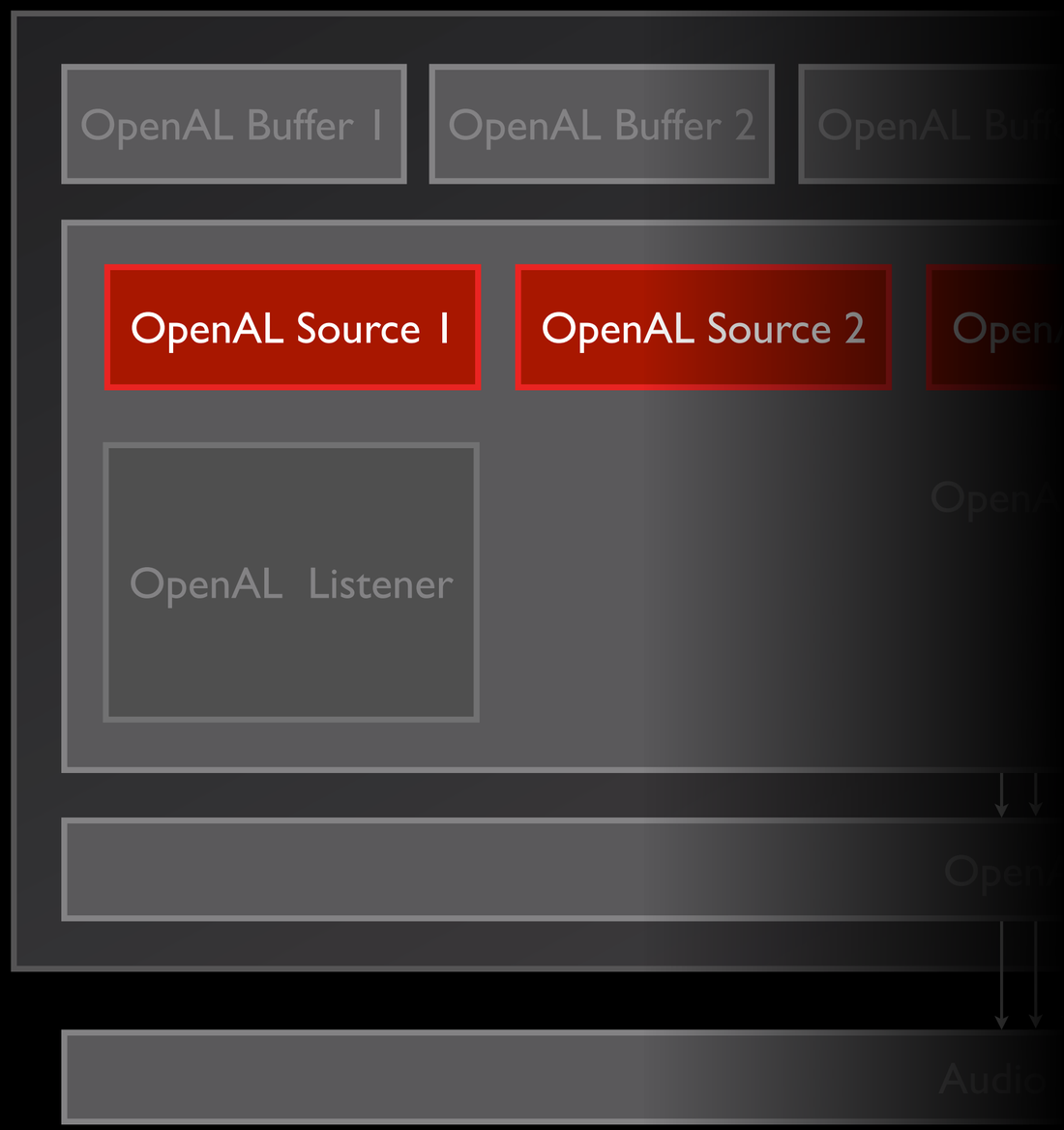




OpenAL Listener: Orientation

x:0, y:0, z:+1 = Listener facing the Positive Z





OpenAL Source

- Sources Make Sound!
- Owned by ONE Context
- Positionable in the Virtual Space
- Many Attributes to Control Rendering

OpenAL Source

Creating/Deleting OpenAL Sources

- ALvoid alGenSources(ALsizei n, ALuint *sources)
- ALvoid alDeleteSources (ALsizei n, ALuint *sources)

```
// Create 5 New OpenAL Sources
ALuint nuSources[5];
alGenSources(5, nuSources); → = {100, 101, 102, 103, 104}

// Delete 2 OpenAL Sources
ALuint deadSources[2] = {100, 102};
alDeleteSources(2, deadSources);
```

OpenAL Source/Listener: Position

AL_POSITION

- Applied to Listener
- Applied to Sources rendering Mono Audio Data
- Defined in World Coordinates. Center = {0.0, 0.0, 0.0}
- Right Handed Cartesian Coordinate System

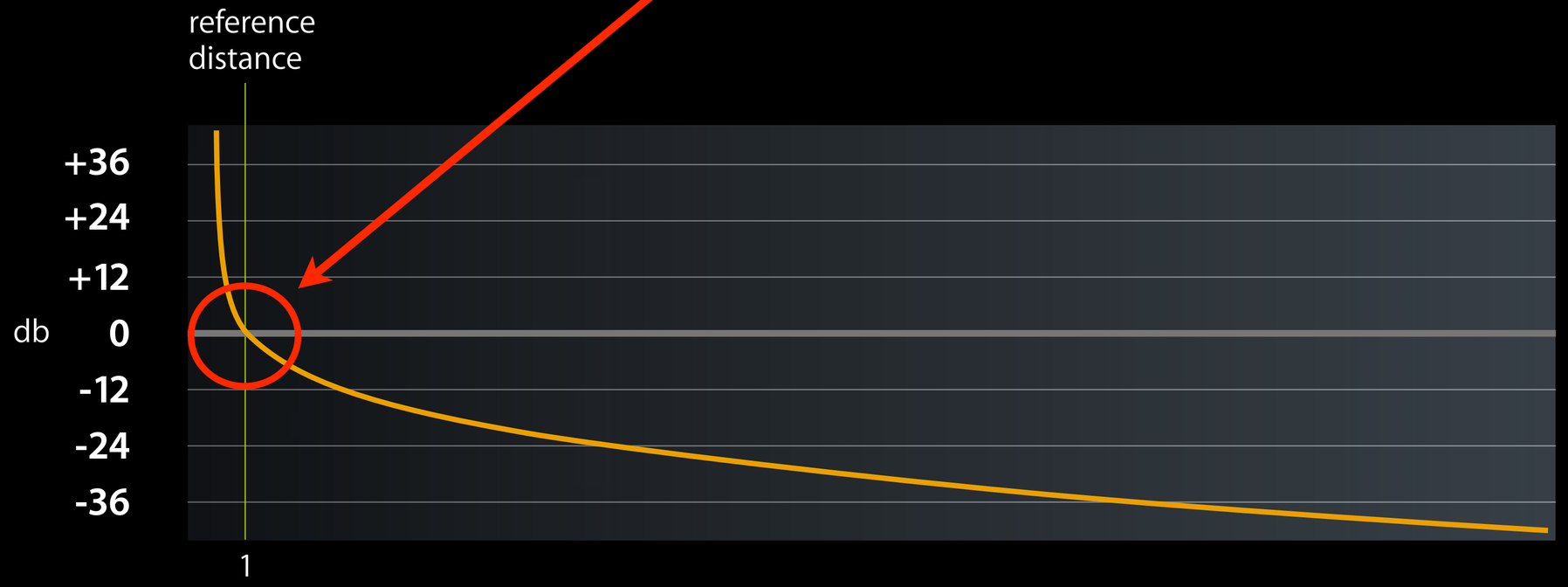
OpenAL: Attenuation by Distance

- Context Settings (Applied to all Sources)
  - Distance Model: `alDistanceModel()`
    - `AL_INVERSE_DISTANCE`
    - `AL_INVERSE_DISTANCE_CLAMPED`
    - `AL_NONE`
- Source Settings
  - `AL_REFERENCE_DISTANCE`
  - `AL_MAX_DISTANCE`
  - `AL_ROLLOFF_FACTOR`
  - `AL_SOURCE_RELATIVE`

OpenAL: Attenuation by Distance

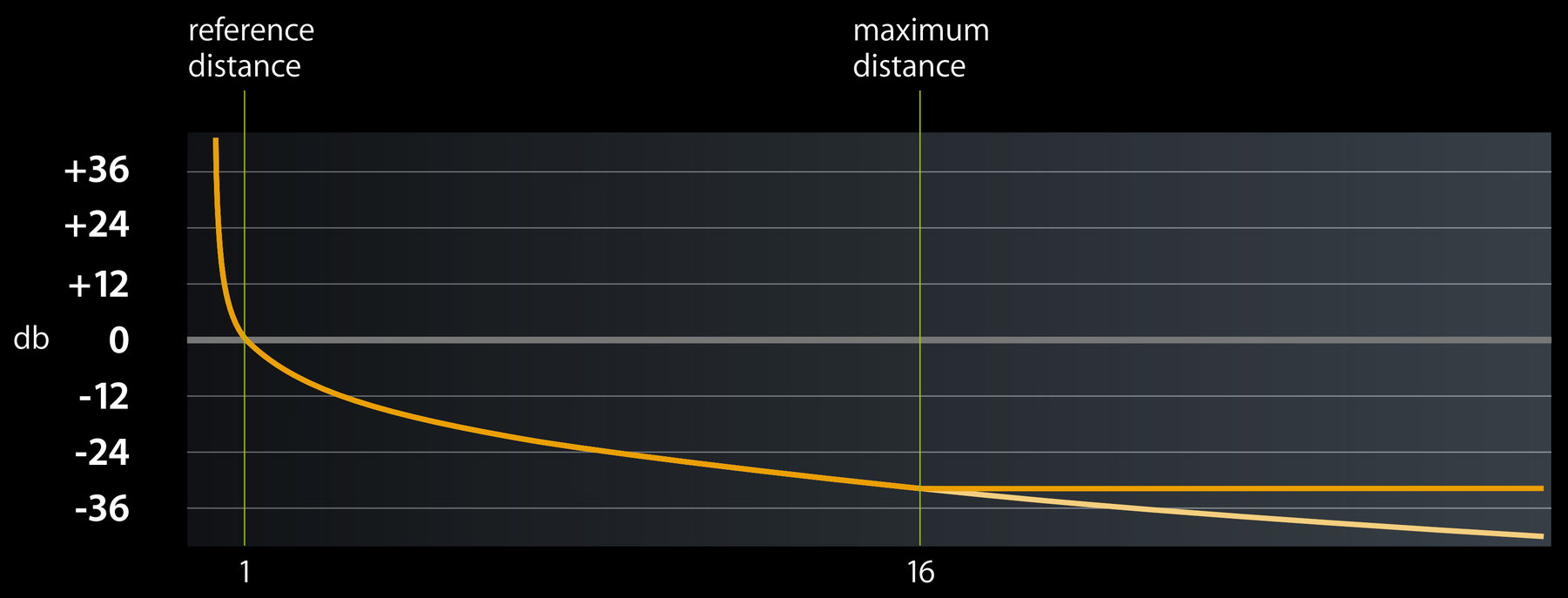
```
// SOURCE SETTING: Get the Source's Reference Distance
ALfloat refDistance;
alGetSourcef(mySource, AL_REFERENCE_DISTANCE, &refDistance);
```

= 1.0



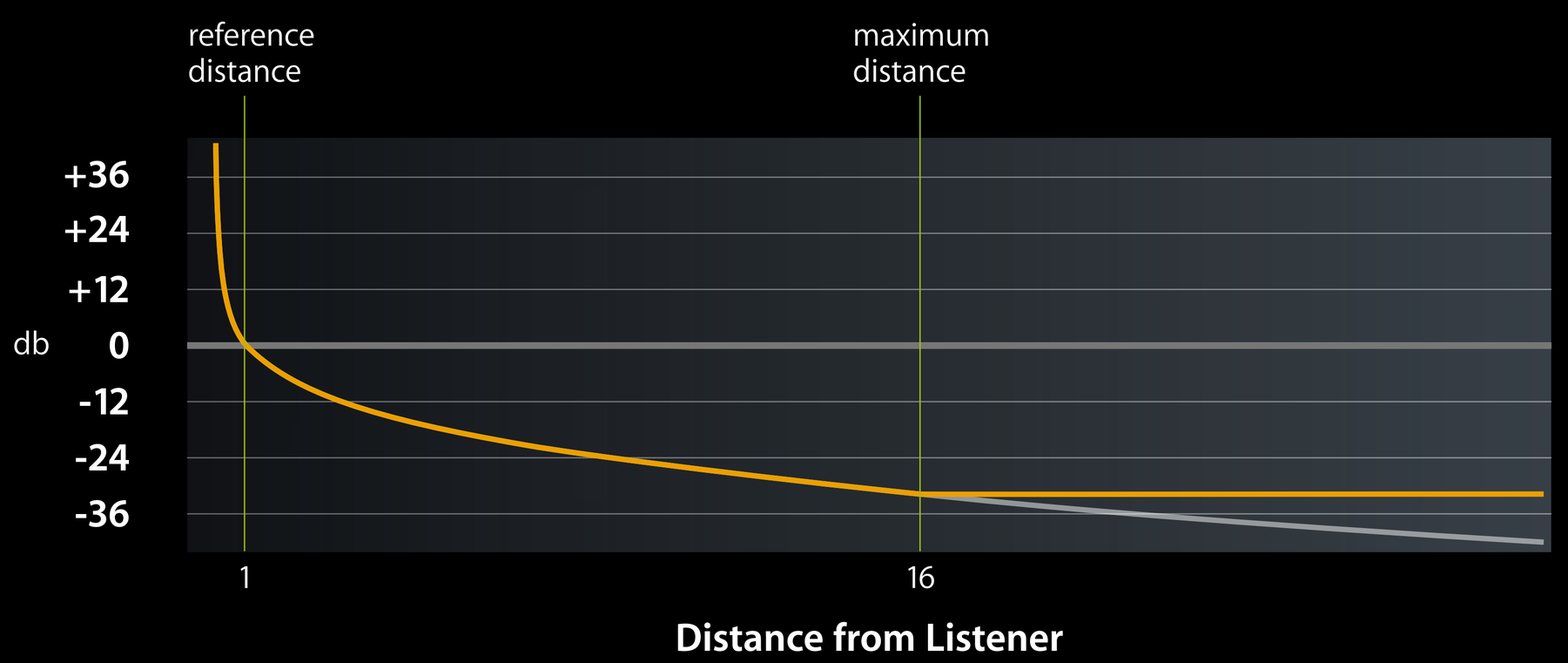
OpenAL: Attenuation by Distance

```
// SOURCE SETTING: Set the Maximum Distance to 16.0
alSourcef(mySource, AL_MAX_DISTANCE, 16.0);
```



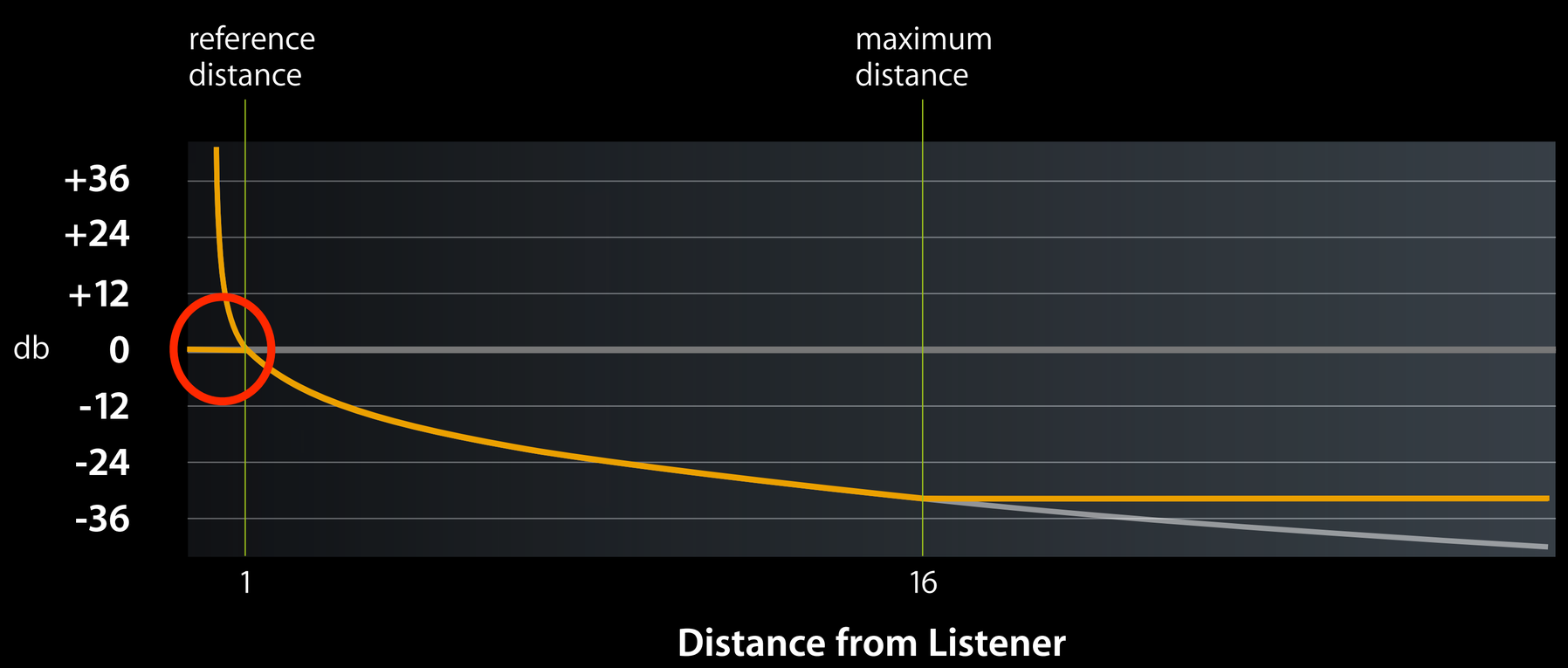
OpenAL: Attenuation by Distance

```
// CONTEXT SETTING: Set the Context Distance Model to AL_INVERSE_DISTANCE_CLAMPED
alDistanceModel(AL_INVERSE_DISTANCE_CLAMPED);
```



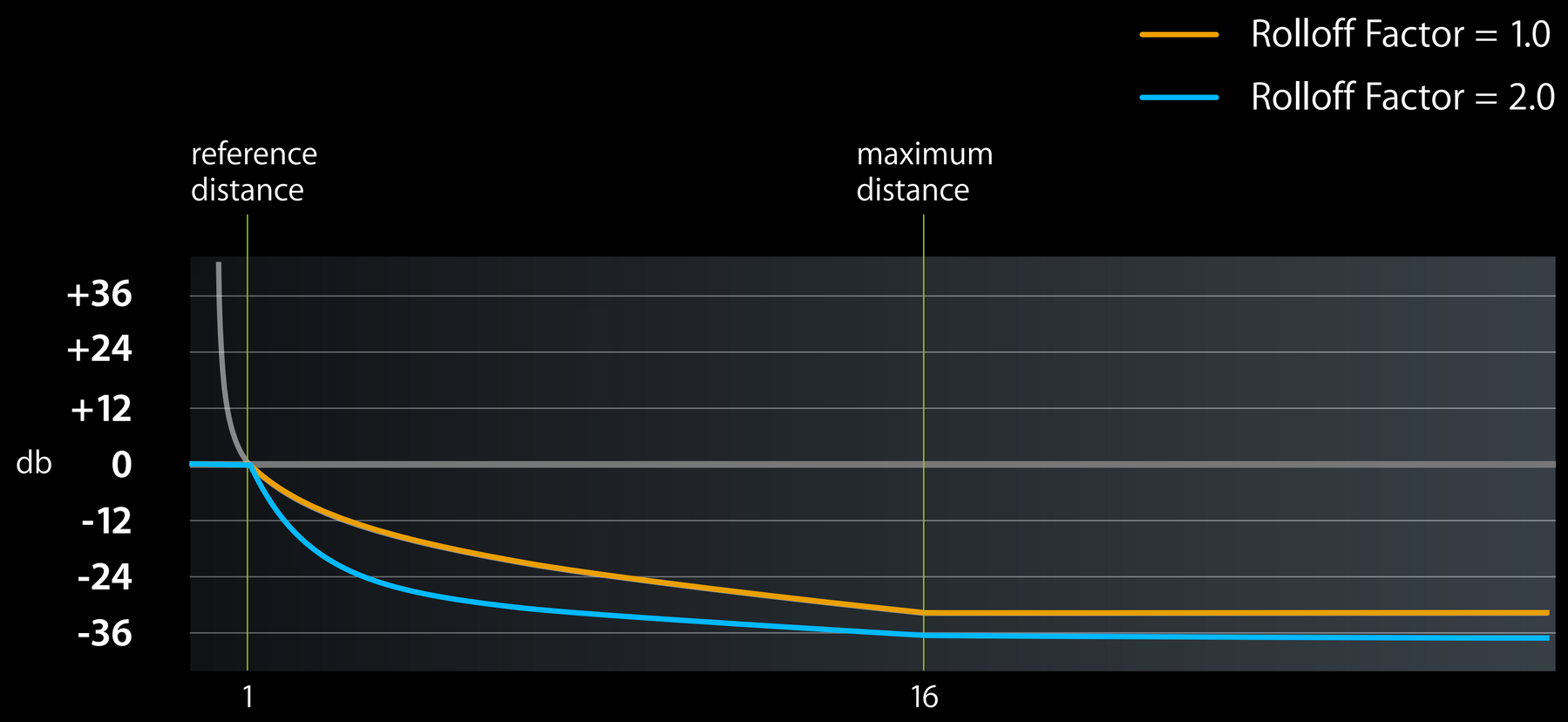
OpenAL: Attenuation by Distance

```
// CONTEXT SETTING: Set the Context Distance Model to AL_INVERSE_DISTANCE_CLAMPED
alDistanceModel(AL_INVERSE_DISTANCE_CLAMPED);
```



OpenAL: Attenuation by Distance

```
// SOURCE SETTING: Change the Source's Rolloff Factor
alSourcef(mySource, AL_ROLLOFF_FACTOR, 2.0);
```



OpenAL Source/Listener: Gain

AL_GAIN

- A Volume Scalar applicable to Source & Listener objects
  - Range: (0.0 - 1.0) = Silence to Full Volume
  - Listener Gain applied to all Context Sources
- `AL_MIN_GAIN / AL_MAX_GAIN:` Source ONLY attribute for bounding gain

```
// Set the Gain of the Source Object to .8
alSourcef(mySource, AL_GAIN, 0.8);
```

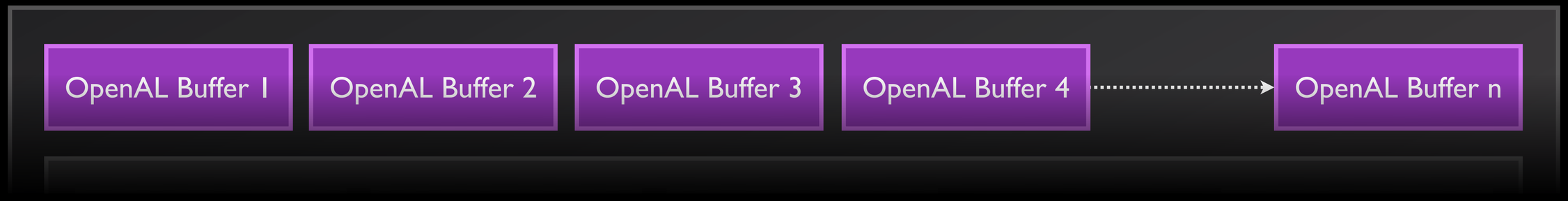
OpenAL Source: Pitch

AL_PITCH

- A Rate Scaler for data rendered by an OpenAL Source
  - 1.0 = NO Pitch Change (default)
  - Settable while rendering

```
// Double the pitch of the source object
alSourcef(mySource, AL_PITCH, 2.0);
```

OpenAL Buffer



- Containers of Audio Data
- Library Makes Copy of the Data
- Reusable
- Shared by ALL Source Objects

OpenAL Buffer

- Creating/Deleting OpenAL Buffers
  - `ALvoid alGenBuffers ( ALsizei n, ALuint *buffers)`
  - `ALvoid alDeleteBuffers ( ALsizei n, ALuint *buffers)`

- Adding Data to an Audio Buffer
  - `ALvoid alBufferData ( ALuint buffer, ALenum format, ALvoid  *data, ALsizei size, ALsizei freq)`

```
AL_FORMAT_MONO8
AL_FORMAT_STEREO8
AL_FORMAT_MONO16
AL_FORMAT_STEREO16
```

Required formats for all implementations

Important Note: Stereo data is ALWAYS rendered to the User's Front Speakers (L,R), regardless of Position or Listener Orientation settings.

OpenAL Source: Managing Buffers

- Adding/Removing OpenAL Buffers
  - `AL_BUFFER`
  - `alSourceQueueBuffers()`
  - `alSourceUnqueueBuffers():`
- Querying Queue State
  - `AL_BUFFERS_PROCESSED`
  - `AL_BUFFERS_QUEUED`
- Looping
  - `AL_LOOPING`

OpenAL Source: Managing the Buffer Queue

```
// Buffer Q already has 2 elements
```

Buffer 1 | Buffer 2

```
// Add 3 more buffers to the Queue
ALuint bufferArray[3] = {bufferOne, bufferThree, bufferFour};
alSourceQueueBuffers(mySource, 3, bufferArray);
```

Buffer 1 | Buffer 2 | Buffer 1 | Buffer 3 | Buffer 4

```
// Check the Q length                                    = 5
ALuint qLength;
alGetSourcei(mySource, AL_BUFFERS_QUEUED, &qLength);
```

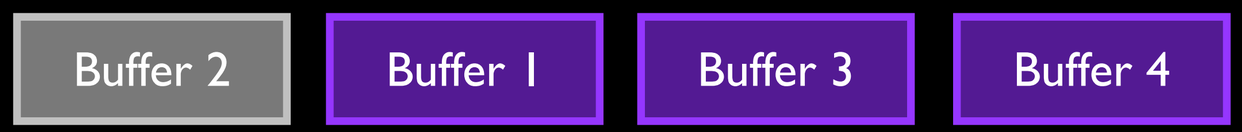
OpenAL Source: Managing the Buffer Queue

```
// check the processed buffers
ALuint processedCount;
alGetSourcei(mySource, AL_BUFFERS_PROCESSED, &processedCount);
```

= 2

```
// remove the 1st buffer in the queue
ALuint bufferArray[1];
alSourceUnqueueBuffers(mySource, 1, bufferArray);
```

bufferArray[0] = bufferOne

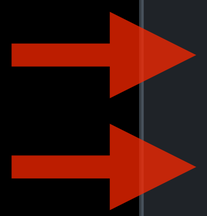


OpenAL Source: Playback Control

- Playing Audio
  - `ALvoid alSourcePlay (ALuint source)`
- Stopping Audio
  - `ALvoid alSourceStop (ALuint source)`
- Pausing Audio
  - `ALvoid alSourcePause (ALuint source)`
  - `ALvoid alSourceResume (ALuint source)`
- Rewinding Audio
  - `ALvoid alSourceRewind (ALuint source)`

Demo Code: Initializing OpenAL, Part 2

```
void InitializeOpenAL()
{
  // Create a new OpenAL Device using the Default Output Device
  ALCdevice  *newDevice = alcOpenDevice(NULL);
  if (newDevice != NULL)
  {
    // Create a new OpenAL Context
    ALCcontext  *newContext = alcCreateContext(newDevice, 0);
    if (newContext != NULL)
    {
      // Make the new context the Current OpenAL Context
      alcMakeContextCurrent(newContext);
      // Create some OpenAL Buffer objects
      alGenBuffers(NUM_BUFFERS_SOURCES, gBuffers);
      // Create some OpenAL Source objects
      alGenSources(NUM_BUFFERS_SOURCES, gSources);
    }
  }
}
```



Demo Code: Filling the Buffers

```
void InitializeBuffers()
{
  ALenum  format;
  ALvoid* data;
  ALsizei size, freq;
  UInt32 i;

  for (i = 0; i < NUM_BUFFERS_SOURCES; i ++)
  {
    // get some audio data from a file
    getAudioFileData(gSourceFile[i], &format, &data, &size, &freq);

    // Attach Audio Data to an OpenAL Buffer
    alBufferData(gBuffer[i], format, data, size, freq);

    // Release the audio data - OpenAL has made it's own copy
    releaseAudioFileData(data);
  }
}
```

Demo Code: Initialize Sources and Play

```
void InitializeSourcesAndPlay()
{
  UInt32   i;
  for (i = 0; i < NUM_BUFFERS_SOURCES; i++)
  {
    // Add a Buffer to the Source's Buffer Q
    alSourcei(gSources[i], AL_BUFFER, gBuffers[i]);
    // Turn Buffer Q Looping ON
    alSourcei(gSources[i], AL_LOOPING, AL_TRUE);
    // Set Position
    alSourcefv(gSources[i], AL_POSITION, gSourcePositions[i]);
    // Set Reference Distance to 5.0
    alSourcef(gSources[i], AL_REFERENCE_DISTANCE, 5.0f);
    // Start Playing the Sound
    alSourcePlay(gSources[i]);
  }
}
```

Demo

https://developer.apple.com/wwdc2005

OpenAL: Extensions

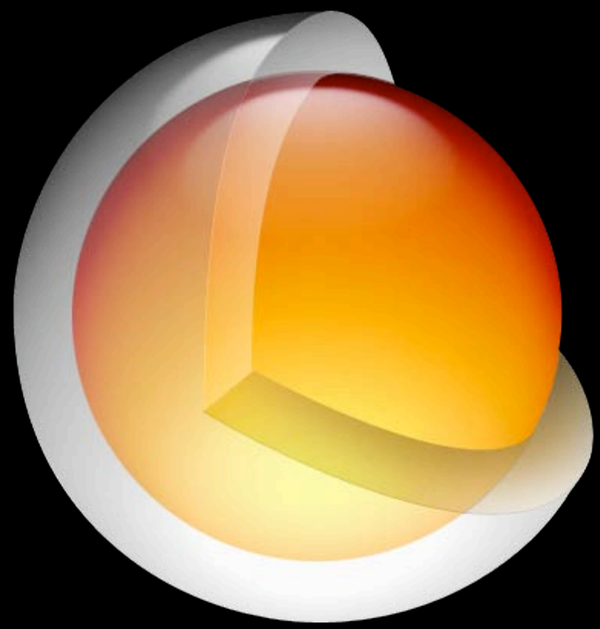
- Augment an implementation with additional features
- Query for Extensions at runtime
- Extension APIs linked dynamically
- Documentation about common Extensions on the OpenAL website

OpenAL: Extensions

- Determine if an Extension is present
  - `boolean alIsExtensionPresent(const ALChar* extName)`
- Get Extension function pointers
  - `void* alGetProcAddress(const ALChar* funcName)`

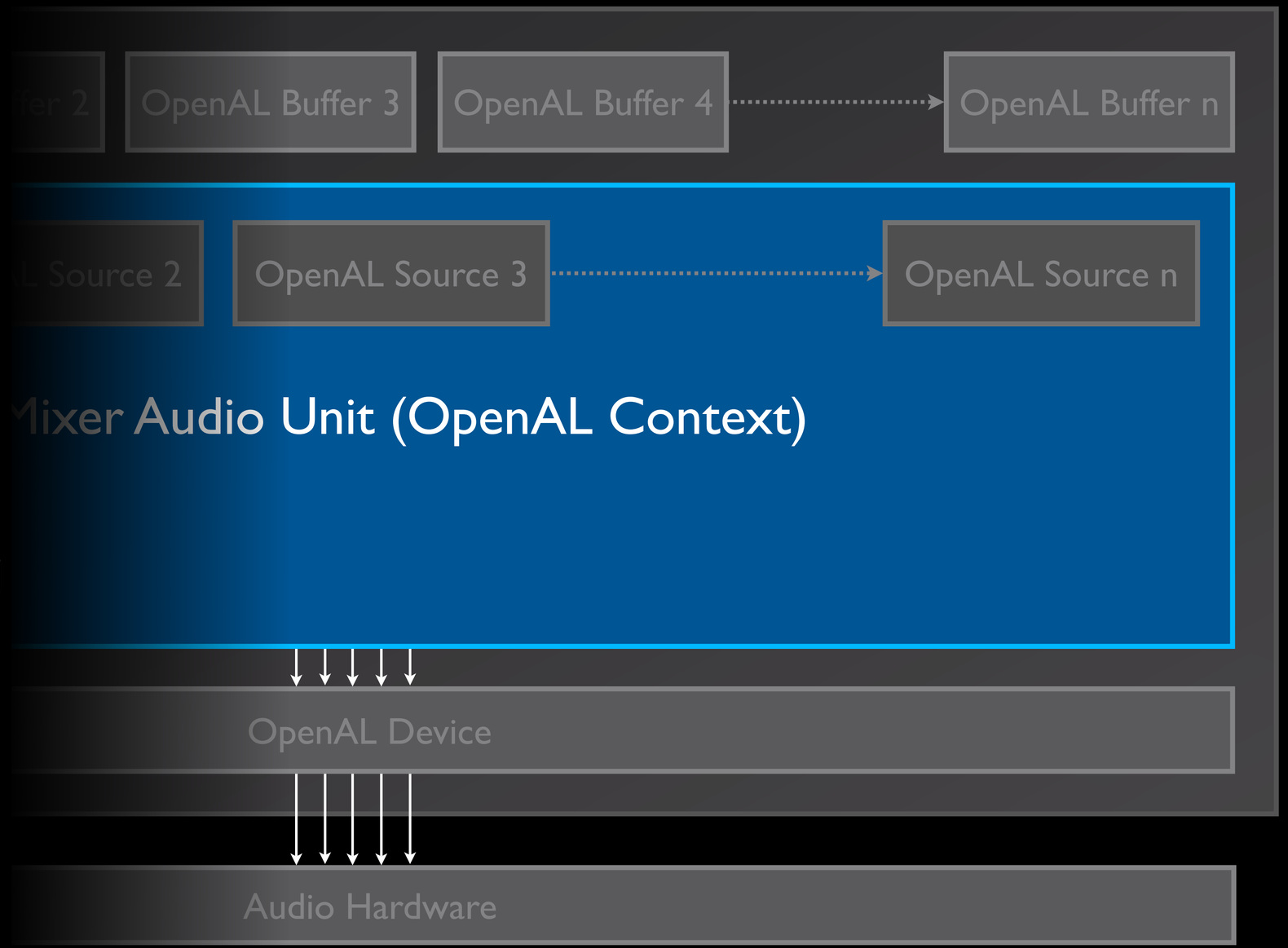
Benefits of Core Audio

- 3DMixer Audio Unit
- Default Output Audio Unit
- AudioConverter APIs
  (AudioToolbox.framework)



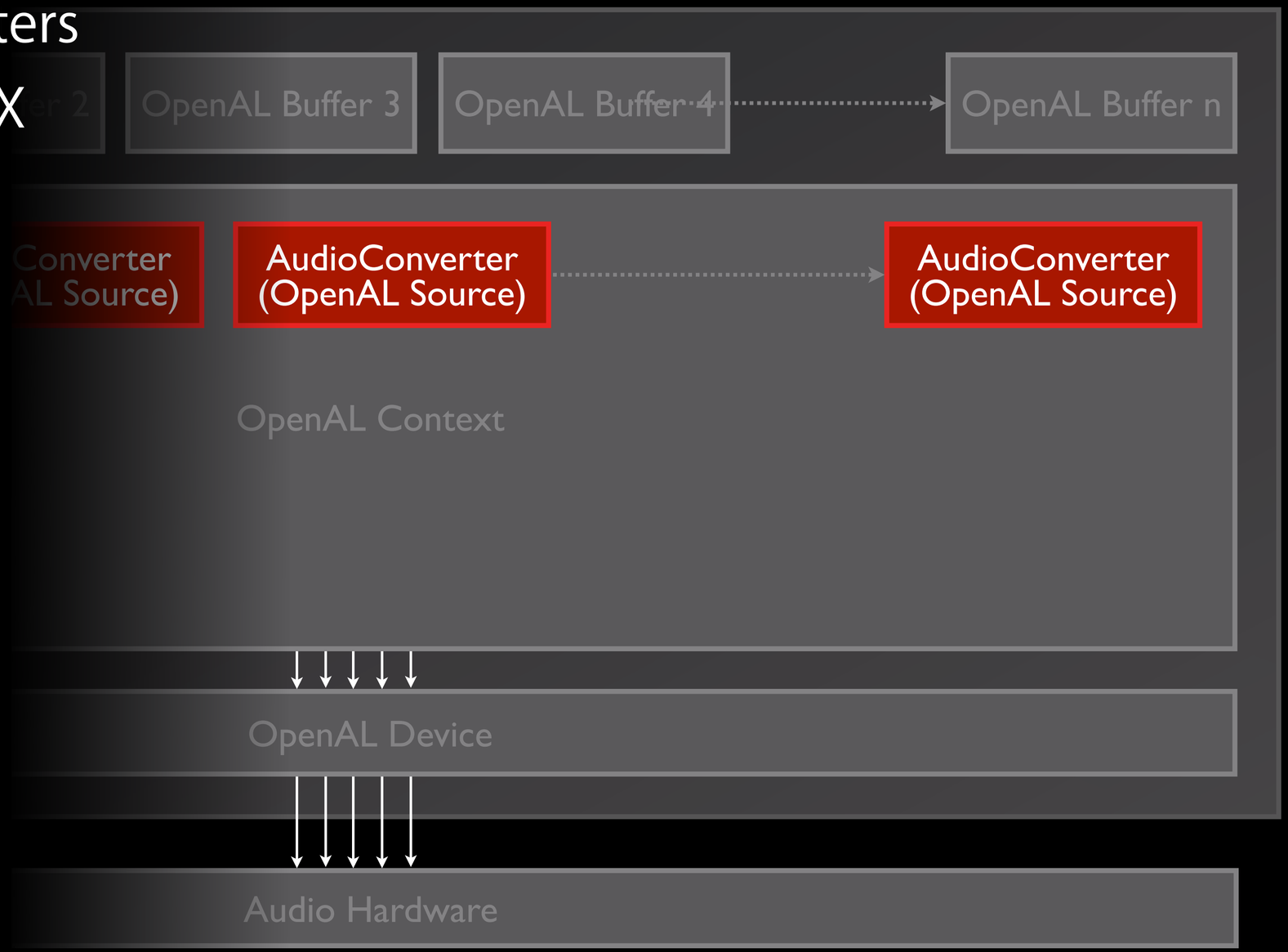
Core Audio: 3DMixer Audio Unit

- Mixing
- Spatial Rendering
- Supported Speaker Configurations
  - Stereo
  - Quadraphonic
  - 5.0
- Optimized Rendering



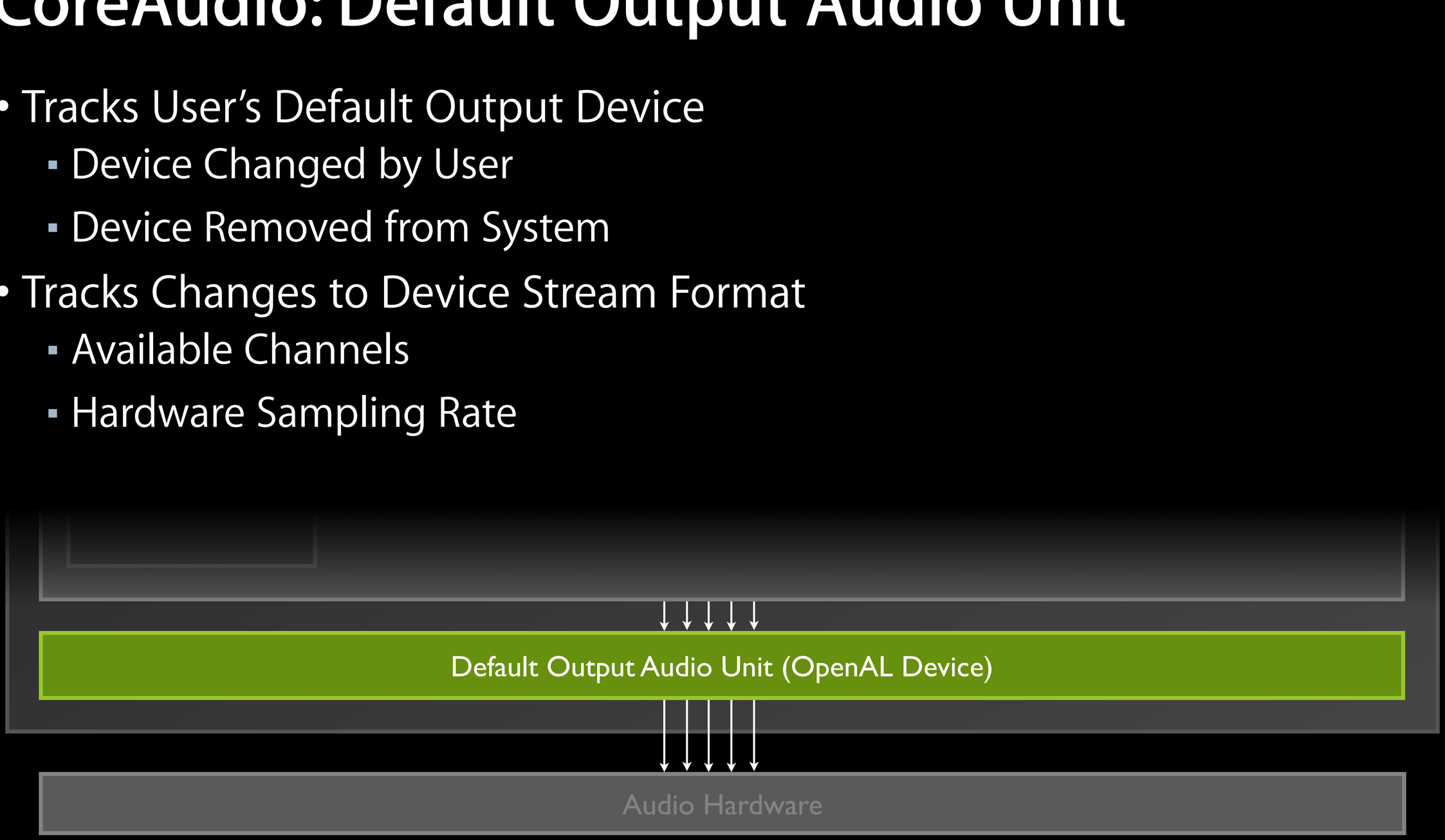
CoreAudio: AudioConverter

- Efficient Int/Float Blitters
- Supports an Mac OS X Supported Format



CoreAudio: Default Output Audio Unit

- Tracks User's Default Output Device
  - Device Changed by User
  - Device Removed from System
- Tracks Changes to Device Stream Format
  - Available Channels
  - Hardware Sampling Rate



Performance Test Results

Rendering 64 Sources (16-Bit PCM) simultaneously

		Non-Apple Implementation	Apple Implementation
G4 (1.3 GHz)	Stereo	24.3%	5.7%
G4 (1.3 GHz)	5.0	Unsupported	8.7%
G5 (2.5 GHz)	Stereo	9.2%	1.8%
G5 (2.5 GHz)	5.0	Unsupported	2.7%

- **% indicates CPU Load**

OpenAL for Mac OS X

Features

- OpenAL Implementation can take advantage of Core Audio features
  - `ALC_SPATIAL_RENDERING_QUALITY`
  - `ALC_RENDER_CHANNEL_COUNT`
  - `ALC_MIXER_OUTPUT_RATE`

OpenAL for Mac OS X: Features

ALC_SPATIAL_RENDERING_QUALITY

- Specify the 3DMixer's Rendering Quality
  - `ALC_SPATIAL_RENDERING_QUALITY_LOW:` Equal Power Panning
  - `ALC_SPATIAL_RENDERING_QUALITY_HIGH:` High Quality HRTF

```
// Use a high quality HRTF for rendering audio data
alSetInteger(ALC_SPATIAL_RENDERING_QUALITY, ALC_SPATIAL_RENDERING_QUALITY_HIGH);

// Get current render quality setting
ALInt   renderQuality = alGetInteger(ALC_SPATIAL_RENDERING_QUALITY);
```

Performance: Using HRTF

		No HRTF	Using HRTF
G4 (1.3 GHz PB)	64 Sources	5.7%	33.8%
	48 Sources		
	32 Sources		
G5 (1.8 GHz IMac)	64 Sources	5.0%	25.7%
	48 Sources		
	32 Sources		
G5 (2.5 GHz DP Tower)	64 Sources	1.8%	11.3%
	48 Sources		
	32 Source		

- % indicates CPU Load

Performance: Using HRTF

		No HRTF	Using HRTF
G4 (1.3 GHz PB)	64 Sources	5.7%	33.8%
	48 Sources	4.3%	24.9%
	32 Sources		
G5 (1.8 GHz IMac)	64 Sources	5.0%	25.7%
	48 Sources	3.8%	19.8%
	32 Sources		
G5 (2.5 GHz DP Tower)	64 Sources	1.8%	11.3%
	48 Sources	1.3%	8.4%
	32 Source		

- % indicates CPU Load

Performance: Using HRTF

		No HRTF	Using HRTF
G4 (1.3 GHz PB)	64 Sources	5.7%	33.8%
	48 Sources	4.3%	24.9%
	32 Sources	2.7%	15.4%
G5 (1.8 GHz IMac)	64 Sources	5.0%	25.7%
	48 Sources	3.8%	19.8%
	32 Sources	2.4%	13.1%
G5 (2.5 GHz DP Tower)	64 Sources	1.8%	11.3%
	48 Sources	1.3%	8.4%
	32 Source	0.95%	5.4%

- % indicates CPU Load

OpenAL for Mac OS X: Features

ALC_RENDER_CHANNEL_COUNT

- Force Rendering to Stereo
  - `ALC_RENDER_CHANNEL_COUNT_STEREO:` Render to stereo regardless of User's multichannel speaker configuration
  - `ALC_RENDER_CHANNEL_COUNT_MULTICHANNEL:` Render to the User's multichannel configuration

```
// Ignore user's multichannel speaker configuration and set rendering to Stereo
alSetInteger(ALC_RENDER_CHANNEL_COUNT, ALC_RENDER_CHANNEL_COUNT_STEREO);

// Get OpenAL's current Render Channel Count setting
ALIntrenderSetting = alGetInteger(ALC_RENDER_CHANNEL_COUNT);
```

OpenAL for Mac OS X: Features

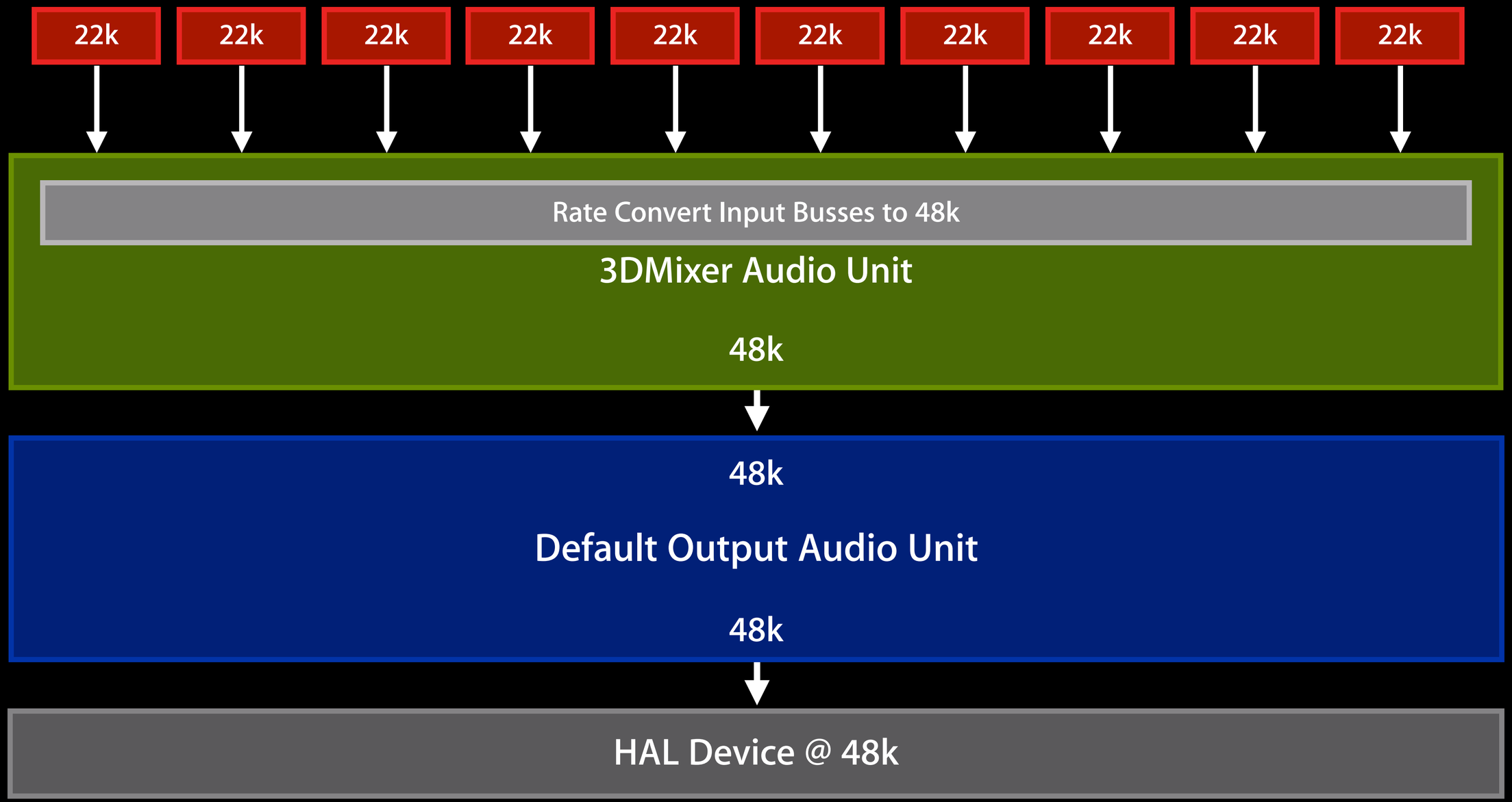
ALC_MIXER_OUTPUT_RATE

- Specify the Sample Rate at which audio data will be mixed

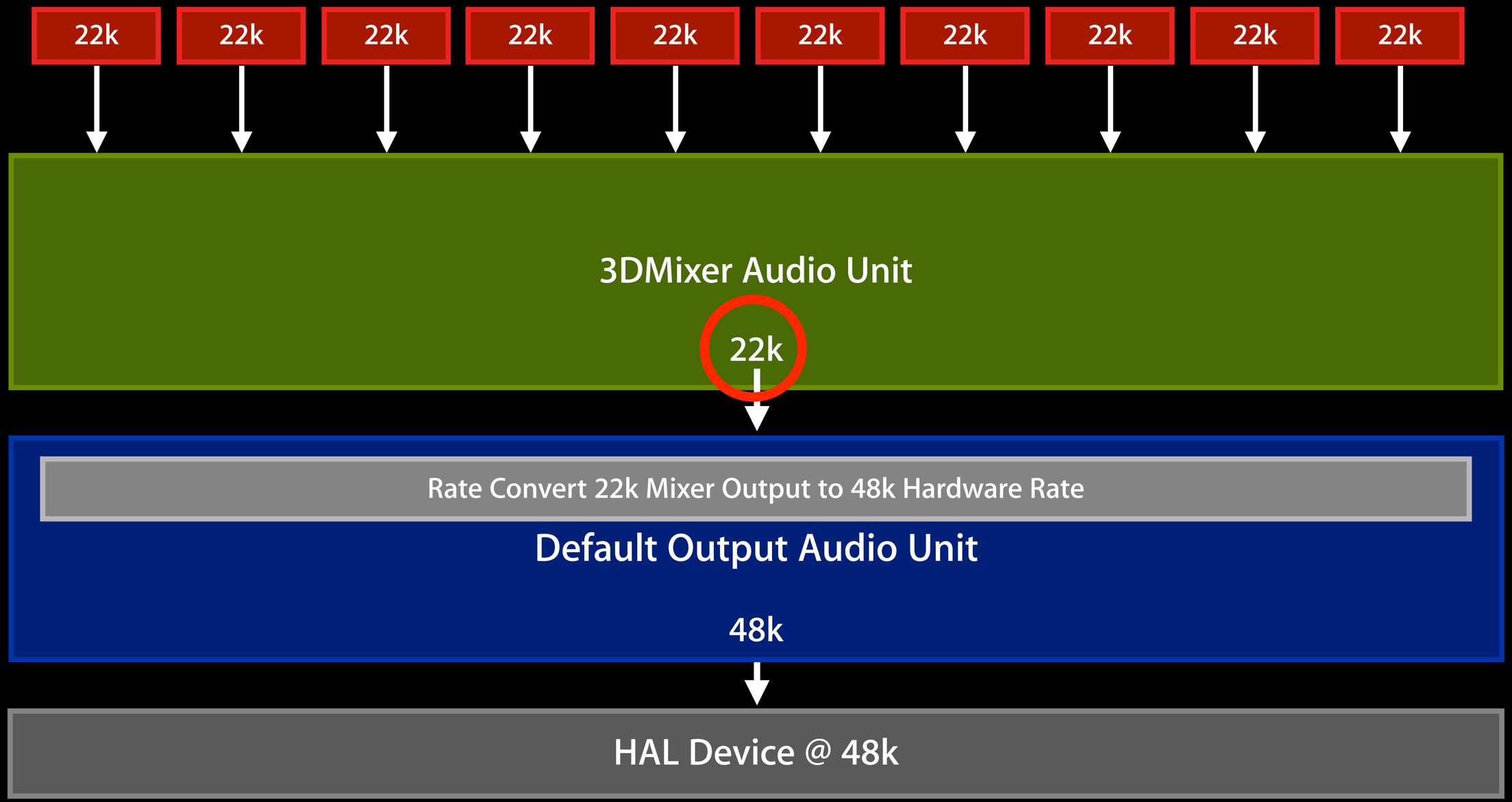
```
// Tell the 3DMixer to mix the data on all input busses at 22k
alSetDouble(ALC_MIXER_OUTPUT_RATE, 22050.0);

// Get current mixer mixing sample rate
ALdouble   mixerOutputRate = alGetDouble(ALC_MIXER_OUTPUT_RATE);
```

OpenAL: Mixing Sample Rate



OpenAL: Mixing Sample Rate



Demo

Doom 3

Glenda Adams
Aspyr Media Inc.

https://developer.apple.com/wwdc2005

More Information

https://developer.apple.com/wwdc2005

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Related Sessions

Core Audio, Audio Units and OpenAL Lab	Graphics and Media Lab Wednesday 12:00PM

Who to Contact

Elliot Sedegah
DTS Engineer
esedegah@apple.com

Rich Hernandez
Marketing Manager Games
rich@apple.com

Bob Aron
Core Audio Engineer
baron@apple.com

Who to Contact: Creative Labs

Garin Hiebert
Administrator
ghiebert@creativelabs.com

OpenAL Website
Specification Documents
CVS Source Repository Links
http://www.openal.org/

OpenAL E-Mail List
http://opensource.creative.com/mailman/listinfo/openal

OpenAL Development E-Mail List
http://opensource.creative.com/mailman/listinfo/openal-devel

Q&A

https://developer.apple.com/wwdc2005

WWDC2005

Worldwide Developers Conference